Power-Aware Replication of Data Structures in Distributed Embedded Real-Time Systems*

Osman S. Unsal, Israel Koren, C. Mani Krishna

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA 01003

Abstract. In this paper, we study the problem of positioning copies of shared data structures to reduce power consumption in real-time systems. Power-constrained real-time systems are of increasing importance in defense, space, and consumer applications. We describe our energy consumption model and present numerical results linking the placement of data structures to energy consumption.

1 System Model

This paper explores the power ramifications of various task assignment heuristics as well as network topology/routing issues. We study distributed real-time systems, with each node having a private memory and each task having a worst-case execution time and deadline. If two tasks reside on different processors then the communication power cost depends on the routing algorithm and topology. The objective is to study the impact of a particular assignment-topology-routing combination on power consumption. To save energy, part of a remote task's data structure may be replicated closer to the consuming node(s). The aim is to find the ideal degree of replication. Increasing the replication increases local memory size and its energy consumption, while decreasing the volume of network transfers and the associated power consumption. Therefore a "sweet spot" may exist, beyond which increasing the degree of replication increases the energy consumption. More formally, the total energy consumed, denoted by E, is:

$$E = \sum_{i=1}^{n\_tasks} E_i \quad \text{where} \tag{1}$$

$$E_i = N_{write_i} \cdot e_{write} + N_{read_i} \cdot e_{read} + \left(\sum_{j=1}^{n\_tasks} N_{net_{ij}} \right) \cdot e_{net} + Size_{mem_i} \cdot e_{static} \tag{2}$$

* This work is supported in part by DARPA through contract No. F30602-96-1-0341. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency, the Air Force or the U.S. Government.

Here, E_i is the energy consumed in executing the i-th task, e_{write} and e_{read} are the memory energy consumption per write and read access, e_{static} is the memory static energy consumption, e_{net} is the energy cost of a per-hop data transfer per-bit, N_{write_i} and N_{read_i} are the number of local memory write and read accesses of task i respectively and $N_{net_{ij}}$ is the number of remote accesses from task i to task j.
If the two tasks i and j are assigned to the same node then, $N_{net_{ij}} = 0$.
Memory consistency is preserved by updating all the replicated copies of a data item when a task writes a shared data item to its private memory. For typical programs the writes are at most 15 percent of reads. This characteristic facilitates the usefulness of replication.
All links are assumed to be of the same type, i.e., the link power consumption to transfer one byte is the same for all links. Various routing strategies such as broadcasting or flooding are also implemented in the model. As for the case of multicasting, efficient multicasting algorithms rely on building and trimming a minimum spanning tree [3]. However, this is not optimal from a power point of view since it builds a minimum spanning tree for all the nodes instead of the subset of nodes in the multicast group. To obtain a better solution, we have developed an energy-saving Steiner tree heuristic for systems with multiple multicasting requirements. Given a weighted graph G, the Steiner tree problem is to find a tree that spans a specified subset of nodes of G with minimal total distance on its edges. Various distinct trees that span the same subset of nodes of G can be constructed and one can select the tree that has less total edge cost than that of the other trees in the set. Since the problem is NP-complete, heuristics are needed. We have adapted such a heuristic [5] for our purposes. The heuristic finds a solution with total distance no more than $2(1 - 1/k)$ times that of the optimal tree in time $O(pn^2)$. Here, n is the number of nodes in G, p the number of Steiner points and k the number of leaves in the optimal Steiner tree. A short description of the Steiner heuristic algorithm is given in Figure 1.
For intertask-communication-bound real-time systems, the allocation of tasks to nodes can also have a significant impact on power. We use a steepest-descent heuristic[2] for power-aware task allocation. The heuristic starts from an initial allocation and then reallocates that pair of tasks to the same node which results in the largest decrease in the energy consumption from among the set of candidate task pairs. This reallocation is done iteratively until the energy saving is below a given threshold. Thus, the heuristic tends to assign tasks which communicate heavily to the same node.

2 Numerical Results

For the results in this section, unless otherwise noted, the number of tasks is 10 and the task execution times, periods as well as intertask communication sizes are random. The number of nodes is 4, the write-to-read power ratio is 1.22, 8% of the memory operations are writes and 92% are reads, and per-hop remote

Step 1. For every multicast group repeat steps 2 through 6.

Step 2. Construct the complete graph H from G and S in such a way that the set of nodes in H is equal to S; for every edge (u, v) in H, the distance of (u, v) is set equal to the shortest path between u and v in G.

Step 3. Find a minimum spanning tree T_H of H.

Step 4. Replace each edge (u, v) in T_H by the shortest path between u and v in G; the resulting graph R is a subgraph of G.

Step 5. Find a minimum spanning tree T_R of R.

Step 6. Delete edges in T_R, if necessary, so that all the leaves in T_R are elements of S. The resulting tree is returned as the solution.

Fig. 1. The Steiner heuristic algorithm

access energy cost is three times that of a local access energy cost.

2.1 Effect of Application Write Ratios

We begin by considering a situation in which each node keeps a fraction of the global data structures in its private memory.
Figure 2 illustrates the effect of changing write ratios on power. As can be seen from the figure the optimum energy point shifts towards lower replication as the write percentage gets higher. Another observation is that as the degree of replication gets higher the energy consumption increases sharply for higher write ratios. This stems from the memory consistency constraint and is caused by the need to update all the replicated copies of a data item that has been modified by the local task.

2.2 Impact of Per-hop Transfer Cost

The per-link power cost per bit transferred depends on the interconnection hardware used. For example, a wireless link may consume less power than a twisted-pair link. If the real-time system designer has multiple options for choosing interconnection hardware, he/she can find the optimum degree of memory replication for each option. Figure 3 illustrates this. Here the per-hop energy consumption varies from being equal to memory energy consumption per operation to four times that of the memory energy consumption per operation. We observe that the optimum energy point shifts toward higher degrees of replication as the per-link power consumption is increased. Also, the system energy consumption starts converging at higher degrees of replication. This phenomenon is due to the fact

that most of the data is locally replicated, thus decreasing the sensitivity of total energy consumption to the per-link power cost.

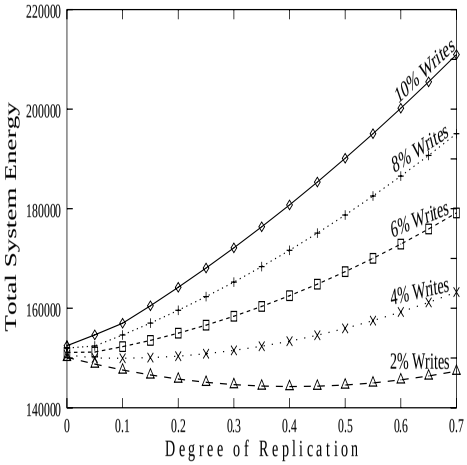

Fig. 2. Effect of write ratios on energy

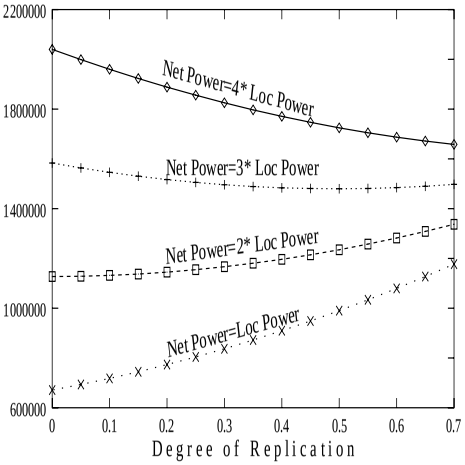

Fig. 3. Effect of per-link power on energy

2.3 Task Allocation and Network Topology

Figure 4 shows the energy consumption impact of the task allocation scheme by comparing the previously mentioned power-aware optimization heuristic with a simple, power-blind round-robin scheme. The resulting saving in energy consumption emphasizes the importance of the task allocation step.

Network topology and routing are also important design considerations in real-time design. Figure 5 shows the energy comparisons of two different choices, a 16-node mesh topology and a 16-node torus topology. The extra wraparound edges of the torus result in lower energy consumption, but the energy difference between the two topologies is not very large.

2.4 Routing Issues

Multicasting has received little attention in real-time systems but is an important problem: sensors providing data to multiple processes and process outputs driving redundant actuators can all benefit from efficient multicasting algorithms. For the baseline case, we implemented the minimal spanning tree truncation scheme [3]. As mentioned in the previous section, we have developed a better multicasting scheme which makes use of a Steiner tree heuristic to find a path with the minimum cost among the multicasting nodes. Figure 6 shows the energy comparison of the two approaches. Here the number of nodes is 16 and the number of tasks is 40.

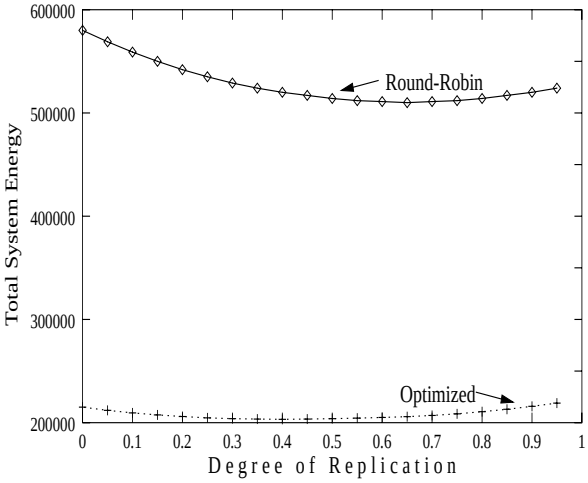

Fig. 4. Impact of task allocation strategy

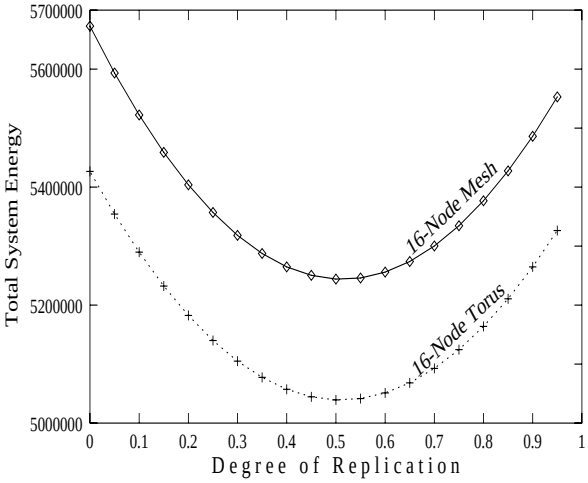

Fig. 5. Topology choice and energy

The routing capabilities of a Real-Time Operating System (RTOS) determines the power impact of multicasting tasks. A minimalist micro-kernel RTOS might just supply a simple flooding model. In this model the multicast message is sent to all the nodes in the system. A slightly more sophisticated RTOS would do a broadcast by sending a unique message to each of the multicast nodes. Broadcasting is considered to be more efficient than flooding [4]. However, as seen in Figure 7, flooding surprisingly does better than broadcasting from an energy point of view. This is because only a single copy of the multicast message is sent in flooding.

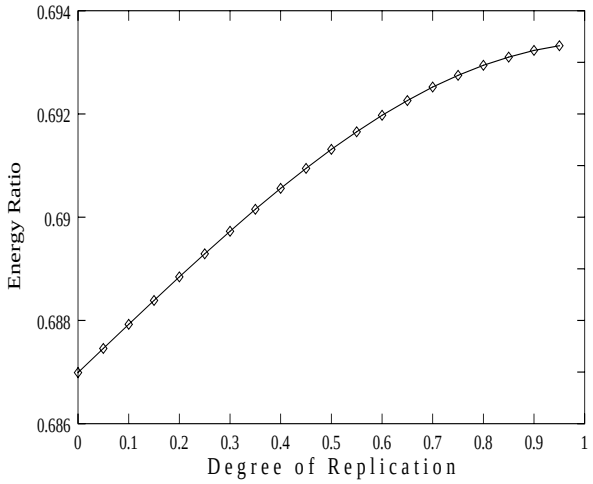

Fig. 6. Energy Ratio of Steiner / Minimal Spanning Tree Truncation

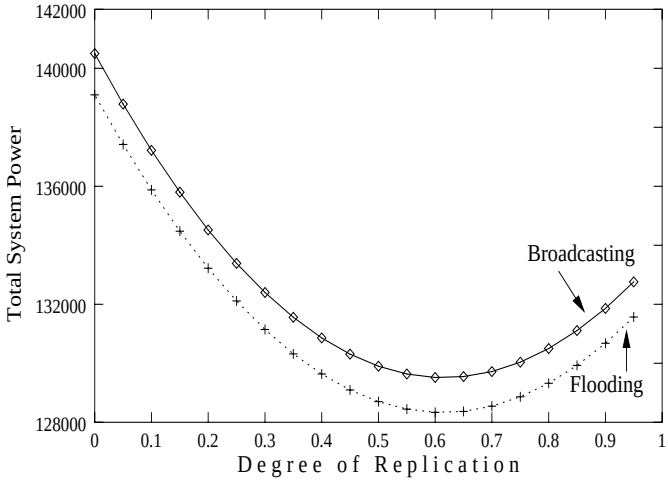

Fig. 7. Flooding versus Broadcasting

2.5 Selective Replication

Up to this point, we have considered the task-to-task communication data structures to be fully replicated. This means that for a multicast group, the data

structure of the multicast source task is replicated at all multicast destination tasks. We now relax this requirement and selectively replicate the data structure of the source task only at some of the destination tasks, thus saving energy. Consider the example of Figure 8. This is a 16-node mesh and part of the task assignment is shown in the figure. Our focus is multicasting group A with task $A.1$ being the source and the other tasks in group A being the destinations. We selectively replicate task $A.1$'s data structure only at task $A.4$'s node. The result is compared against full replication and no replication in Figure 9. Here, the energy is plotted against the per-hop energy cost and it is normalized with respect to the energy consumption of no replication. As can be seen, selective replication results in significant energy savings.

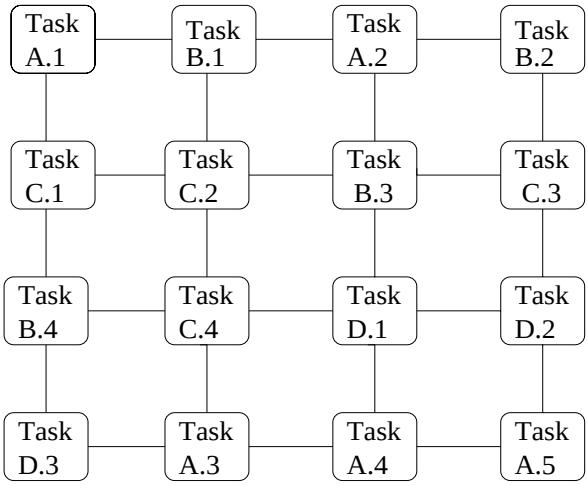

Fig. 8. Example for selective replication

3 Conclusion

We have constructed a model to gauge the power impact of task assignment, network topology and routing strategies within the context of data structure replication to decrease energy. Our results show that substantial energy savings are possible by careful design.

Our model also gives us the ability to calculate the energy impact of new power aware heuristics. We have adapted a Steiner tree heuristic for multicasting and compared its energy consumption with the baseline case of minimal spanning tree truncation.

Currently, we are studying the more general case of heterogeneous data consumption rates at the destination tasks. We are also developing a heuristic which will

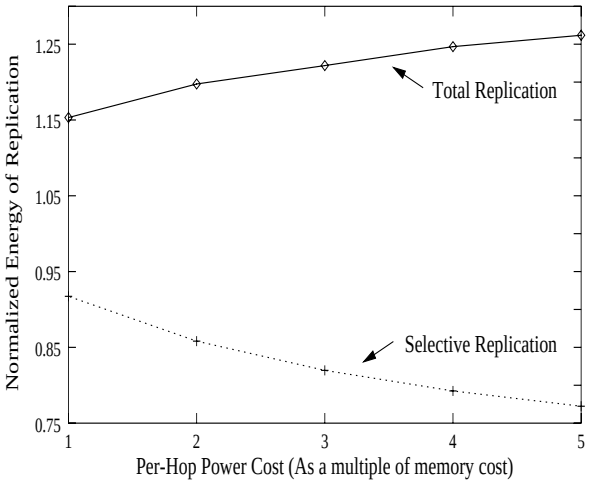

Fig. 9. Advantage of selective replication

optimize the memory replication needs of each task.

References

1. Coumeri, S. L., and Thomas, D. E., Memory Modeling for System Synthesis, www.ece.cmu.edu:80/ thomas/research/List.html
2. Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., Numerical Recipes, Cambridge University Press, 1989
3. Deering, S. E., and Cheriton, D.R., Multicast Routing in Datagram Internetworks and Extended LANs, ACM Trans. on Computer Systems May 1990
4. Tanenbaum, A. S., Computer Networks, Third Edition, Prentice Hall, 1996
5. Lau, H. T., Combinatorial Heuristic Algorithms with FORTRAN , Springer-Verlag, 1986

This article was processed using the LaTeX macro package with LLNCS style